

Konzept für ein computerbasiertes Wirtschaftspiel

Projektarbeit November 2004 – Februar 2005

Studiengang: Kommunikation und Informatik

Studenten:
Rieser Micha
Scrugli Giancarlo

Betreuender Dozent:
Prof. Dr. Hans-Peter Hutter

1 Inhaltsverzeichnis

2 Abbildungsverzeichnis

3 Abstract

In früheren Projekt- und Diplomarbeiten wurde ein generisches Planspiel entwickelt. Mit diesem ist es möglich, Planspiele nach dem Prinzip „Tragödie der Allmende" einfach zu entwickeln. Wir haben dieses Framework untersucht auf seine Tauglichkeit zur Entwicklung komplexerer Planspiele. Es hat sich dabei sehr rasch gezeigt, dass der dort verwendete Ansatz, das Spiel in einer einzigen Finite State Maschine (kurz: FSM) abzubilden, auf komplexere Planspiele nicht übertragbar ist.

Wir haben Planspiele aller Art untersucht und versucht einen gemeinsamen Nenner zu finden. Wir haben festgestellt, dass es bei vielen Planspielen um direkte zwischenmenschliche Kommunikation geht und diese Spiele deshalb gar nie implementierbar sind. Es handelt sich um Spiele, die Rheotorik, Teambildung, Teamdynamik etc. betreffen. Solche Spiele leben von der Kreativität der Mitspieler und der grenzenlosen Möglichkeiten während der Spielrunde. Wir mussten deshalb zuerst herausfinden, welche Planspiele grundsätzlich implementierbar sind. Wir mussten uns auch von Simulationen abgrenzen. Simulationen wollen einen Ausschnitt aus der Wirklichkeit möglichst detailreich und realitätsgetreu abbilden. Bei Planspielen steht aber der Lerneffekt im Vordergrund und werden deshalb für diesen Lerneffekt entworfen. Sie bilden die Wirklichkeit nur soweit ab, wie dies überhaupt notwendig für den Lernprozess ist.

Für unser Konzept haben wir schlussendlich das Brettspiel Ökolopoly gewählt. In diesem Planspiel geht es um die Steuerung der Volkswirtschaft eines Landes. Es kann mit mehreren Spielern oder alleine gespielt werden. Es existieren von diesem Spiel bereits computerbasierte Implementierungen.

Unser Konzept für ein komplexeres Wirtschaftspiel knüpft an die letzte Diplomarbeit von Schmid/Stutzer an. Wir erweitern die Idee der Diplomarbeit, indem wir anstatt nur mit einer FSM, mit mehreren verschiedenen FSMs das Planspiel zu modellieren versuchen. Die Automaten werden miteinander verknüpft, so dass der Ausgang eines Automaten zum Eingang eines anderen geführt wird. Weil sie in einer definierten Reihenfolge nacheinander ablaufen müssen, mussten wir für unser Konzept noch eine Verlaufssteuerung entwerfen. Das Problem der Steuerung haben wir über ein Petrinetz gelöst. Das Petrinetz steht über den verknüpften FSMs und steuert so den Ablauf. Wir haben das Petrinetz in Java implementiert.

Mit unserer Arbeit haben wir den Grundstein für die Implementierung von komplexeren Planspielen, wie Ökolopoly und Stratagem, gelegt.

4 Aufgabenstellung

Projektarbeit 1

Studiengang:	Kommunikation und Informatik	Jahr:	2004
Studierende:	Micha Rieser, Giancarlo Scrugli		
Klasse:	KI3d	Dozentin/Dozent:	Dr. H.-P. Hutter
Ausgabe der Aufgabe:	15. November 2004	Unterschrift:	
Abgabetermin:	18. Februar 2004		

Thema: Konzept für ein computerbasiertes Wirtschaftsspiel

In früheren Projekt- und Diplomarbeiten wurde ein generisches Planspiel entwickelt. Mit diesem ist es möglich, verschiedene Planspiele nach dem Prinzip "Tragödie der Allmende" einfach zu entwickeln. In der vorliegenden Projektarbeit soll zuerst untersucht werden, wie gut sich dieses Framework für die Entwicklung von komplexeren Planspielen (z.B. Wirtschaftspiele) eignet. Sodann soll das bestehende Konzept so erweitert werden, dass auch komplexere Planspiele einfach entwickelt werden können. Die eigentliche Umsetzung des Konzepts ist für eine spätere Projekt- oder Diplomarbeit vorgesehen.

Aufgaben

1. Stellen Sie eine Liste von Wirtschaftspielen zusammen, die bereits computerbasiert sind oder sich für eine Implementation auf dem Computer eignen. Vergleichen Sie diese Wirtschaftspiele und finden Sie heraus, ob und welche gemeinsame Basis diese Spiele haben.
2. Stellen Sie anhand Ihrer Erkenntnisse aus der ersten Teilaufgabe die Anforderungen an ein generisches Planspiel zusammen, mit dem man einfach auch komplexere Planspiele, insbesondere Wirtschaftsspiele, entwickeln kann.
3. Entwickeln Sie ausgehend vom vorhandenen Planspielkonzept mindestens ein Konzept für ein Framework, das die Anforderungen aus Teilaufgabe 2 erfüllt.

Quellen

[SchSt04] E. Schmid, T. Stutzer: *Generisches Planspiel* , Diplomarbeit ZHW, Studiengang KI, Nov. 2004.

[Ulrich04] M. Ulrich: *Mit Planspielen nachhaltige Wirkung erleben*, 2004. www.ucs.ch/service/download/docs/artikelpsnaha.pdf

[Ulrich02] M. Ulrich: *Sind Planspiele langwierig und kompliziert*, 2002. www.vernetzt-denken.de/PDF_Dokumente/5_01.pdf

5 Einleitung

Unsere Aufgabe bestand darin, Planspiele zu analysieren, ein generisches Konzept dafür zu entwerfen und ein erstes Framework zu erstellen.

Wir haben nun erfahren, dass die Planspiele, die wir näher betrachten haben, viel mit Kybernetik zu tun haben. Kybernetik ist die Wissenschaft von der Struktur komplexer Systeme. Vor allem geht es bei der Kybernetik um das Zusammenspiel zwischen Teilsystemen und den Regelungsmechanismen im Gesamtsystem. Die Kybernetik beschreibt deshalb alle möglichen komplexen Systeme wie Gesellschaftssysteme, ökologische Systeme, Robotik oder auch biologische Systeme. Der Mensch selber ist wohl das komplexeste kybernetische System.

Bei vielen Planspielen geht es nun genau darum, die Abhängigkeiten der Teilsysteme im Gesamtsystem zu entdecken und auch zu erkennen, wie sich das Verändern von Beziehungen zwischen den Einzelsystemen auf das Gesamtsystem auswirkt. Der Spieler soll somit seinen Blickwinkel erweitern, dass er nicht nur Einzelsysteme betrachtet und nur diese zu steuern versucht, was meistens katastrophale Folgen hat, sondern das Gesamtsystem als solches begreift und deshalb für das Gesamtsystem die richtigen Entscheidungen fällen kann. Dies wird dann als „vernetztes Denken" bezeichnet.

Der kürzlich verschiedene Prof. Dr. Dr. h.c. Frederic Vester war Mitglied des „Club of Rome" und leitete die „Studiengruppe für Biologie und Umwelt" der Universität München. Er war Fachmann in Umweltfragen und des „vernetzen Denkens". Er erfand 1983 das Spiel Ökolopoly, welches spielerisch die kybernetischen Vernetzungen innerhalb eines Gesellschaftssystems aufzeigt. Dieses Planspiel ist für unsere Arbeit das Referenzmodell. Es erfüllt nämlich die Anforderung an ein komplexeres Planspiel und ist der Vorgänger für viele andere Planspiele in den Bereichen Ökologie, Wirtschaft oder Kommunikation. Wir werden in dieser Dokumentation immer wieder auf dieses Spiel zurückkommen und zeigen damit schrittweise auf, wie wir dieses Spielprinzip generisch umsetzen wollen.

Von der ganzen Arbeit ging für uns ein hoher Reiz aus, da im Verlaufe des Studiums Kommunikation und Informatik an der ZHW häufig auf das vernetzte Denken zurückgegriffen wurde und wir es gezielt angewendet haben. Z.B. war es Thema im Fach „Kultur, Gesellschaft und Sprache", in der MTU-Woche und im Fach „Kommunikation".

5.1 Motivation

Wir möchten mit diesem Konzept einen Kern für weitere Arbeiten schaffen. Das Konzept soll zum Erstellen von computerbasierten Planspielen verwendet werden können.
Das bestehende Konzept aus den vorgängigen Projekt- und Semesterarbeit wollen wir erweitern, um noch „generischer" zu werden.

5.2 Das Planspiel

5.2.1 Definition

„Ein Planspiel versetzt die TeilnehmerInnen in eine fiktive Situation. Diese ist ein vereinfachtes Abbild der Realität. Während mehreren Spielrunden machen sich die TeilnehmerInnen mit der Situation vertraut, führen Verhandlungen und fällen konkrete Entscheide. Daraus ermittelt das (Computer-) Modell die Ausgangslage für die nächste Spielrunde.
Die anschliessende Transferphase verbindet die Erfahrungen aus dem Planspiel mit der Realität. So erwerben die TeilnehmerInnen Erfahrungen und praxisnahes Handlungswissen."[1]

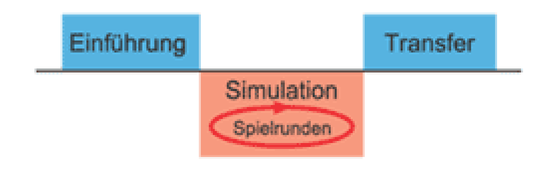


Abb. 1: Die drei Phasen eines Planspiels

„Definitionen, was ein Planspiel sei, gibt es viele. Die traditionelle Definition besagt, daß mehrere Spielparteien gegen- oder miteinander spielen, aber nur über die Spielleitung Kontakt miteinander haben. Alle Spielzüge werden schriftlich der Spielleitung abgegeben, von dieser ausgewertet, bzw. weitergeleitet und dokumentiert.Planspiele haben eine sehr lange Geschichte, und wurden hauptsächlich in der militärischen Ausbildung eingesetzt. Seit den 70er Jahren gibt es verstärkt Planspiele für die Bildungsarbeit, mit denen soziales Verhalten, Konfliktlösungsverhalten etc. geprobt werden kann. In den letzten 10 Jahren ist, zumindest in der Bildungsarbeit, das traditionelle Schema etwas aufgebrochen worden und es gibt sehr viele Planspiele, die der obigen Definition nicht mehr entsprechen.Planspiele haben immer eine große Nähe zum Simulationsspiel, allerdings steht beim Planspiel eher der zu bearbeitende Konflikt, bzw. die beteiligten Personen (Rollen) im Vordergrund."[2]

---
[1] Ulrich, über Planspiele
[2] TU Darmstadt

5.2.2 Lernziele eines Planspiels

Ein Planspiel verfolgt unter anderem folgende Lernziele:

- Entscheidungsfindung unter Zeitdruck
- bereits bestehendes Wissen anwenden und vertiefen
- fördern von vernetztem, ganzheitlichen Denken
- Umgang mit komplexeren Problemen trainieren
- kybernetische Zusammenhänge transparent machen
- Wirkung von Entscheidungen risikolos erfahren

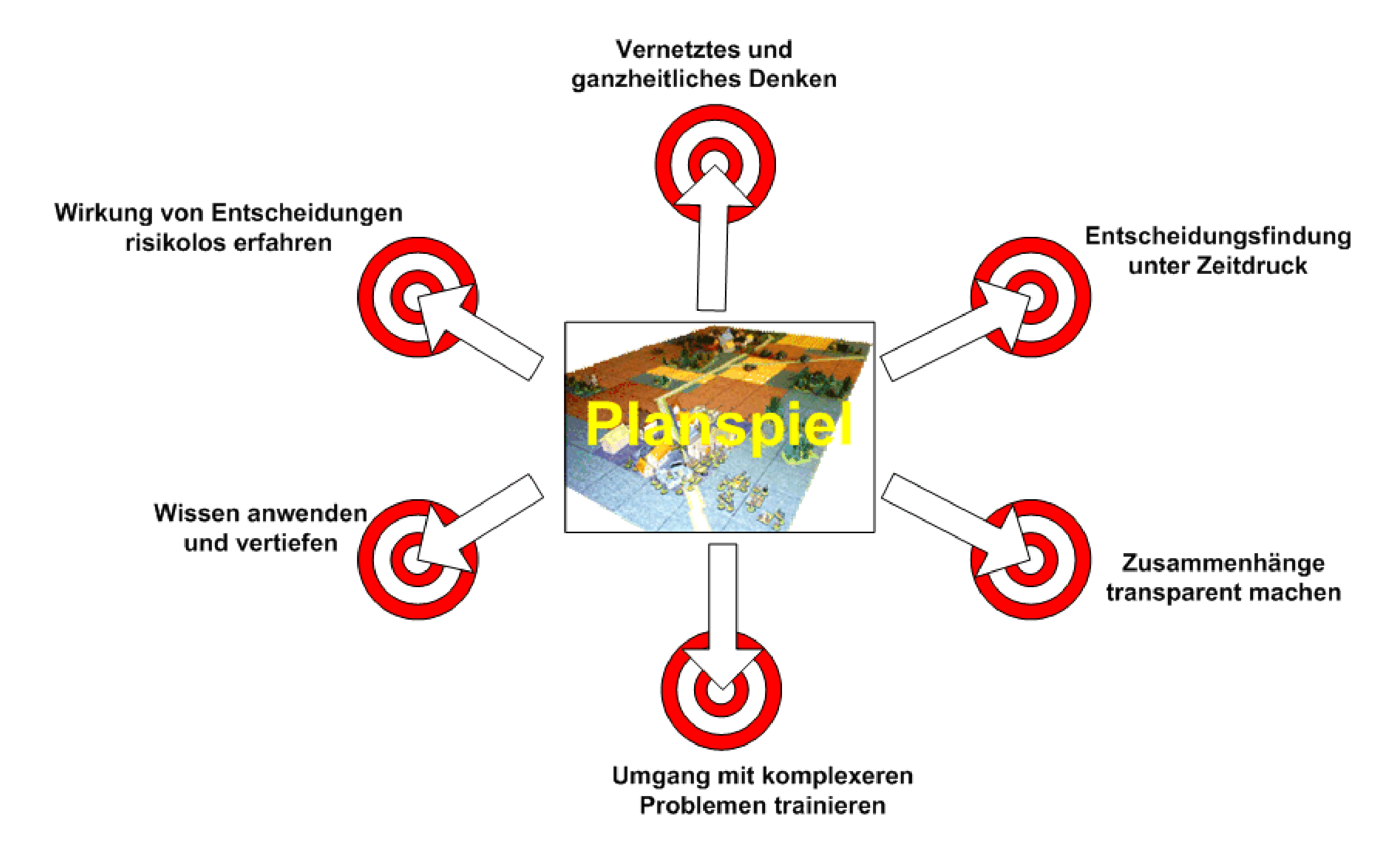


Abb. 2: Ziele von Planspielen

5.2.3 Vergleich verschiedener Planspiele

Wir haben die für uns wichtigen Aspekte von mehreren Planspielen hier zusammengetragen.

NCG(New Commons Game):

Beschreibung	Das Spiel thematisiert das Spannungsfeld zwischen kurzfristiger Profitmaximierung und langfristiger Erhaltung einer Ressource.	
Schwerpunkte	Das NEW COMMONS GAME erlaubt verschiedene Schwerpunkte (Strategien zur Nutzung natürlicher Ressourcen, Problematik freier Güter, Allmendeproblematik, Verhalten in Teams und Organisationen).	
Eckdaten:	Spieler Anzahl	6-24
	Dauer	1h – 6h
	Computerbasiert	☑ (computergestützt)
	Rundenbasiertes Spiel	☑
Quelle	UCS Ulrich Creative Simulations Dr. Markus Ulrich	

STRATAGEM:

Beschreibung	Das Spiel vermittelt System- und Sozialkompetenz. Die TeilnehmerInnen gestalten die Volkswirtschaft und die ökologische Situation eines Landes.	
Schwerpunkte	-Systemkompetenz: Kompetent Handeln in vernetzten Systemen unter Einbezug sachlicher und sozialer Faktoren. -Datenanalyse (Datenauswertung, Zusammenführen der Einzelinformationen im Team, Entscheidungsfindung, Interpretation der Computerresultate). -Volkswirtschaftliche Grundlagen und Zusammenhänge verstehen.	
Eckdaten:	Spieler Anzahl	4-8 Teilnehmende je Land
	Dauer	½ - 2 d
	Computerbasiert	☒
	Rundenbasiertes Spiel	☑
Quelle	UCS Ulrich Creative Simulations Dr. Markus Ulrich	

Das Landwirtschaftsspiel:

Beschreibung	Thema im Spiel ist das Spannungsfeld der wirtschaftlichen Konkurrenz und ökologischer Anforderungen.	
Schwerpunkte	Das verallgemeinerte Spielkonzept ist geeignet für interaktive, wissenschaftlich basierte Darstellung von Landschaftsveränderungen.	
Eckdaten	Spieler Anzahl	1
	Dauer	5 – 30min.
	Computerbasiert	✓
	Rundenbasiertes Spiel	✗
Quelle	UCS Ulrich Creative Simulations Dr. Markus Ulrich	

COREBIFA®:

Beschreibung	Das Spiel stellt den Ernährungssektor von Landwirtschaft bis KonsumentInnen dar.	
Schwerpunkte	Praxisnahe Vertiefung von Volks- und Betriebswirtschaft. Nachhaltige Entwicklung im Zeitraffer erleben und mitgestalten. Category & Product Management (Lebensmittelhandel)	
Eckdaten	Spieler Anzahl	10-25
	Dauer	1 - 2d
	Computerbasiert	✗
	Rundenbasiertes Spiel	✓
Quelle	UCS Ulrich Creative Simulations Dr. Markus Ulrich	

HEX:

Beschreibung	Hier stehen Kommunikation, Zusammenarbeit und effiziente Mittelverteilung innerhalb eines Landes, bzw. einer Organisation im Zentrum.	
Schwerpunkte	Kommunikation in komplexen Organisationen. Training von sozialer Kompetenz und Teamfähigkeit. Kooperation und Kommunikation als Schlüsselfaktor für nachhaltige Entwicklung trainieren.	
Eckdaten	Spieler Anzahl	10 - 16
	Dauer	½ - 1 d
	Computerbasiert	✗
	Rundenbasiertes Spiel	✓
Quelle	UCS Ulrich Creative Simulations Dr. Markus Ulrich	

Ökolopoly:

Beschreibung	Zur Steuerung einer Gesellschaft bekommen die Spielenden "Aktionspunkte" zugeteilt, die sie als Ressourcen in vier von acht Lebensbereichen investieren können: Sanierung, Produktion, Aufklärung, Lebensqualität (die anderen, nur indirekt beeinflussbaren Bereiche laufen unter den Kürzeln "Politik", "Bevölkerung", "Umwelt" und "Vermehrungsrate").	
Schwerpunkte	Die Spielenden können in den internen Bau von Ökolopoly vollständig Einsicht nehmen, also ermitteln, von welchen Annahmen das Modell ausgeht und wie es die Zusammenhänge zwischen den Faktoren definiert.	
Eckdaten	Spieler Anzahl	1 (auch mehr)
	Dauer	max. 2h
	Computerbasiert	✗
	Rundenbasiertes Spiel	✓
Quelle	Ravensburger	

5.2.4 Spielbeschreibung Ökolopoly

Ökolopoly ist ein Planspiel, welches die Abhängigkeiten verschiedener Bereiche des menschlichen Lebensraumes simuliert. Die Idee dahinter ist, dass der Benutzer lernt, nicht die Anforderungen und Wirkungsweisen einzelner Systeme zu betrachten, sondern das Gesamtnetzwerk der Abhängigkeiten zu verstehen.

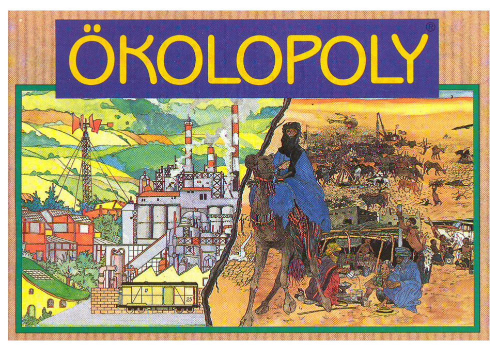
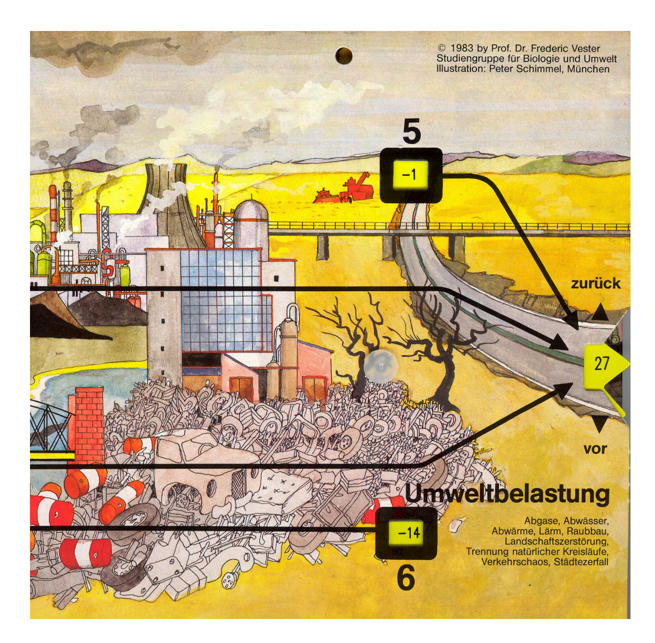


Abb. 3

Im Ökolopoly stellt der Benutzer die Regierung des Landes „Kybernetien" (Industriestaat) oder „Kyborien" (Nomadenvolk) dar. Acht Lebensbereiche sind in diesem Planspiel in beiden Fällen verknüpft. Die Bereiche sind in Drehscheiben versteckt.



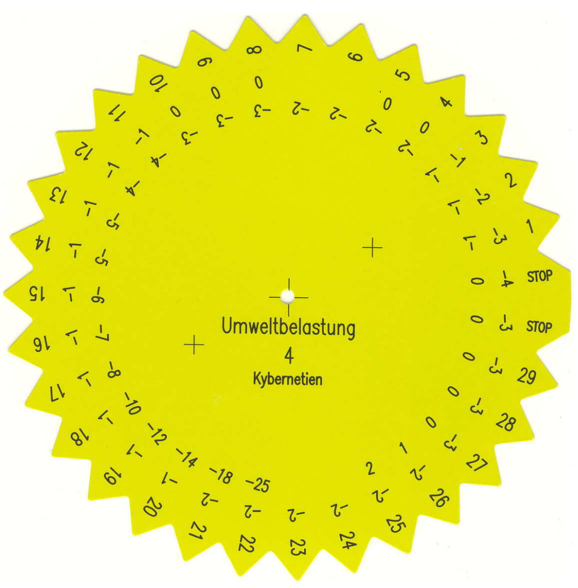


Abb. 4/5

Im Spiel verteilt der Benutzer dann die Aktionspunkte so auf die Bereiche „Produktion", „Lebensqualität", „Aufklärung" und „Sanierung", dass die Beziehungen dieser Bereiche das Land in ein stabiles Gleichgewicht überführten sollen. Der Erfolg zeigt sich an der „Politik", die eine Note für den Spieler darstellt.

Wichtig ist, dass der Benutzer aktiv mitverfolgt, welche Kette von Wirkungen und Rückwirkungen Entscheidungen auf das ausgewählte Land haben. Es fördert somit aktiv das vernetzte Denken des Benutzers.

Aus den Scheiben lassen sich die einzelnen Punkte herauslesen und graphisch darstellen. Bsp. bei der Scheibe der *Umweltbelastung* (Abb. 5), lesen wir aus dem Fenster 6 die *Wirkung auf die Lebensqualität* heraus.

Zustand	Wirkung	Zustand	Wirkung
1	2	16	-3
2	1	17	-3
3	0	18	-4
4	0	19	-4
5	0	20	-5
6	0	21	-5
7	0	22	-6
8	-1	23	-7
9	-1	24	-8
10	-1	25	-10
11	-2	26	-12
12	-2	27	-14
13	-2	28	-18
14	-2	29	-25
15	-3		

Daraus lassen sich dann mit einer Tabellenkalkulation Kurven erstellen:

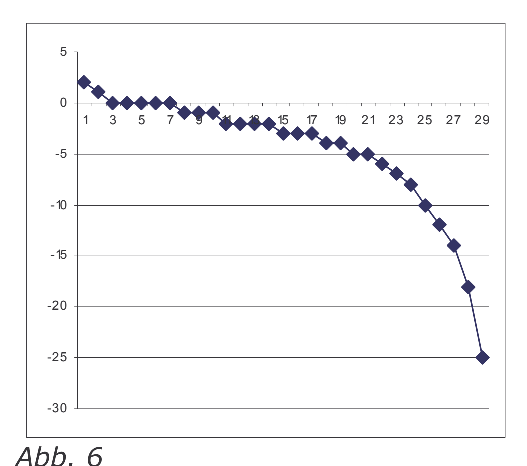


Abb. 6

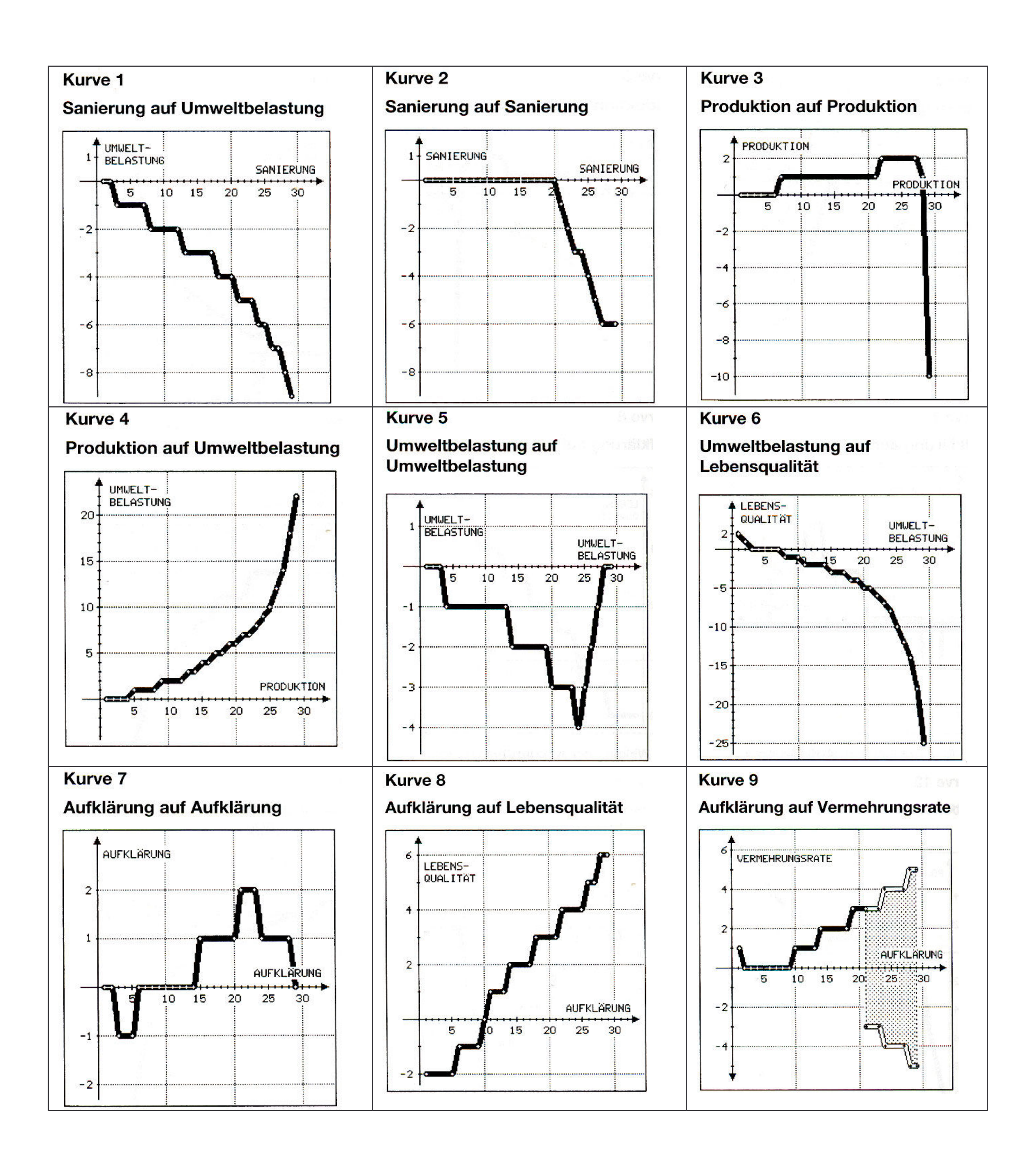
Hier sehen wir, nun die Tabelle oben in graphischer Form.

Auf der X-Achse wird der Zustand der Umweltverschmutzung dargestellt.

Auf der Y-Achse lässt sich dann die Wirkung auf die Lebensqualität ablesen.

Es handelt sich also um eine Funktion Y = f(X).

Graphische Darstellungen über alle Wirkungen, wie sie im Ökolopoly vorkommen:


Kurve 1
Sanierung auf Umweltbelastung
UMWELT-
BELASTUNG
SANIERUNG
Kurve 2
Sanierung auf Sanierung
SANIERUNG
SANIERUNG
Kurve 3
Produktion auf Produktion
PRODUKTION
PRODUKTION
Kurve 4
Produktion auf Umweltbelastung
UMWELT-
BELASTUNG
PRODUKTION
Kurve 5
Umweltbelastung auf
Umweltbelastung
UMWELT-
BELASTUNG
UMWELT-
BELASTUNG
Kurve 6
Umweltbelastung auf
Lebensqualität
LEBENS-
QUALITÄT
UMWELT-
BELASTUNG
Kurve 7
Aufklärung auf Aufklärung
AUFKLÄRUNG
AUFKLÄRUNG
Kurve 8
Aufklärung auf Lebensqualität
LEBENS-
QUALITÄT
AUFKLÄRUNG
Kurve 9
Aufklärung auf Vermehrungsrate
VERMEHRUNGSRATE
AUFKLÄRUNG

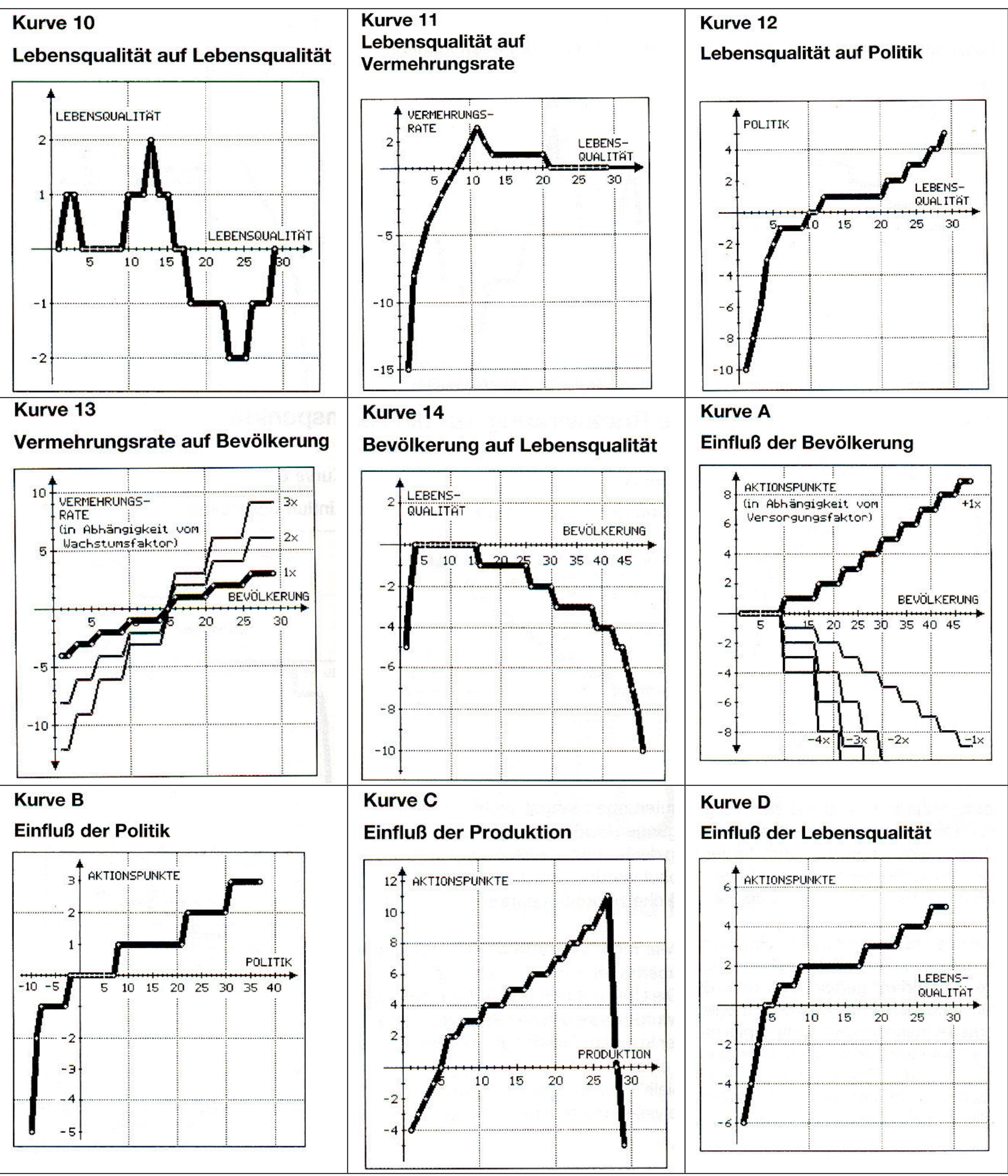


Abb. 7-24

In jeder Spielrunde muss der Benutzer die folgenden Punkte durchlaufen:

- *Rundenzähler einstellen:* Vor der ersten Runde wird der Rundenzähler auf "Start" gestellt, in jeder weiteren Runde wird der Zähler um einen Schritt weitergedreht und die in den Fenstern erscheinenden Angaben befolgt, was meistens in eine Ziehung einer Aktionskarte mündet, wo weitere Spielangaben drin stehen.

- *Aktionspunkte verteilen:* In dieser Phase des Spiels wird nun entschieden, wie die verfügbaren Aktionspunkte eingesetzt werden sollen. Nun müssen die zur Verfügung stehenden Aktionspunkte auf die im Spielplan angegebenen vier Bereiche „Sanierung", „Produktion", „Aufklärung" und „Lebensqualität" verteilt werden. Dazu dreht der Benutzer die Drehscheiben um entsprechend viele Schritte vor und die Scheibe "Aktionspunkte" entgegengesetzt so viele Schritte zurück. Der Bereich "Produktion" ist der einzige, der auch zurückgedreht werden kann. Auch das kostet Aktionspunkte.

- *Verfolgen der kybernetischen Wirkungen:* Sobald die Aktionspunkte gesetzt sind, läuft eine Kette von Aktionen und Reaktionen ab, die vom Benutzer beobachtet werden muss. Er muss nämlich in der Reihenfolge der nummerierten Fenster (1 - 14) Pfeile entlangzufahren und die entsprechenden Scheiben um entsprechend viele Schritte vor- oder zurückzudrehen. Dabei wird er sequentiell alle Scheiben in einer vordefinierten Reihenfolge abarbeiten. Es kommt natürlich vor, dass er eine Scheibe zwei- oder dreimal bewegen muss.

- *Vorbereiten der nächsten Runde:* Zunächst wird das neue Entscheidungspotential (Aktionspunkte) ermittelt. Dieses ergibt sich aus der Summe der Fenster A-D.

- *Die nächste Runde beginnt.*

Eine wichtige Variante des Spieles:

Man kann nun einen grösseren Lerneffekt erreichen, wenn man mit anderen Spielern eine "Regierung" bildet und Ressorts auf die Gruppe verteilt. Die Aktionspunkte verwaltet dann der Regierungschef und einige Finanzminister, die „Produktion" vertritt ein Wirtschaftsminister, die „Sanierung" ein Umweltminister, die „Aufklärung" ein Kulturminister, die „Lebensqualität" der Gesundheitsminister und die „Bevölkerung" der Familienminister.

Der Regierungschef muss seine Punktevergabe überzeugend gegenüber den anderen Spielern vertreten können, denn letztlich entscheidet nicht er, sondern die Mehrheit der Regierung. Wie in Demokratien üblich, wird der Regierungschef nach vier oder fünf Jahren (Runden) neu gewählt. Der alte Regierungschef kann auch wieder gewählt werden, wenn die „Politik" mit zu den Aktionspunkten beigetragen hat (über acht Aktionspunkte).

Natürlich kann eine Gruppe von Spielern auch alle Entscheidungen und Aktionen gemeinsam beschließen und ausführen.

Diese Variante des Spiels funktioniert nun sehr ähnlich, wie das Planspiel Stratagem. Das ist der Grund, warum wir Ökolopoly als unser Modellspiel

betrachtet haben. Wenn wir in unserem Konzept das Ökolopoly abbilden können, dann haben wir auch die Voraussetzung für Planspiele wie Stratagem geschaffen, die heutzutage noch häufig in der Praxis eingesetzt werden. Das Konzept muss aber gleichzeitig auch einfacher implementierbare Planspiele wie die „Tragödie der Allmende" umfassen. Erst dann haben wir ein generischeres Konzept als die PA/DA von Schmid/Stutzer erreicht.

5.3 Vorgängige Arbeiten

In einer früheren Projekt- und Diplomarbeiten wurde von Schmid/Stutzer ein generisches Planspiel entwickelt. Mit diesem Konzept ist es möglich Planspiele nach dem Prinzip „Tragödie der Allmende" zu entwickeln. Grundsätzlich haben sie das ganze Planspiel in einer einzelnen Finite State Machine mit mehreren Ein- und Ausgängen abgebildet. (Jede Spielgruppe stellt ein eigener Ein- und Ausgang dar.) Bei der FSM handelt es sich um einen Moore-Automaten. D.h. der Eingang ändert den Zustand und nur abhängig vom Zustand wird ein Ausgangswert gebildet. Die eigentliche Spiellogik wird dann in den Funktionen des Automaten abgebildet.

6 Vorgehensweise

Wir sind von der Diplomarbeit von Schmid/Stutzer von einer Finite State Machine ausgegangen. Das Spiel „New Commons Game" funktioniert wie ein endlicher Zustandsautomat. Die Spieler sind verantwortlich für verschiedene Eingaben (blaue Karte, rote Karte, grüne Karte, gelbe Karte oder orange Karte). Aufgrund dieser Eingaben verändert sich der Zustand der Ressource. Die Ausgabe ist dann der Ertrag an Fischen, der jede Spielergruppe bekommt. Als Computerspiel auf einem verteilten System kann man es als FSM lösen, wo die Eingaben irgendwo zentral gesammelt werden, der Zustand sich dann ändert und die Ausgaben dann wieder auf die Rechner verteilt werden muss.

Um einen generischen Ansatz für komplexere Spiele zu erhalten, haben wir versucht herauszufinden, ob wir nicht grundsätzlich alle Planspiele, die sich auf Computer implementieren lassen, auf eine einzelne FSM abbilden können. Rein theoretisch ist das möglich, da auch sehr komplexe Simulationen ja auf einem Rechner laufen, der auch nichts anderes ist als eine Finite State Machine. Dabei stellen die erreichbaren Zustände die möglichen Kombinationen der Bits in einem RAM dar. Uns machte aber ziemlich bald Mühe, komplexe Planspiele abzubilden, weil die Zustände unüberblickbar wurden. Es handelt sich um eine kominatorische Explosion von Zuständen. Bei unserem Modellspiel Ökolopoly können wir die Anzahl der möglichen Zustände sehr einfach berechnen: Die Anzahl ist die Multiplikation aller möglichen Zustände der Drehscheiben. Es handelt sich somit um 48,3 Billionen (36x29x29x29x29x29x29x48x47) verschiedene Zustände. Es wird nun klar, dass wir dies unmöglich in einer einzelnen FSM abbilden können, wo wir alle Zustände in einer Liste beschreiben müssen.

Uns fiel dann das Machiavelli-Prinzip „divide et impera" ein. Wir können jede Scheibe des Ökolopoly als einzelne FSM ansehen. Wir haben somit das Problem auf neun Finite State Machines mit höchstens 48 Zuständen herunter gebrochen. Die einzelnen Statemachines sind im Prinzip sehr einfach gehalten und wir waren uns sicher, dass wir mit unserem Prinzip immer noch den generischen Ansatz der PA/DA Schmid/Stutzer erfüllen. Anstatt mehr Zustände haben wir nun mehr FSMs.

Die Frage, die uns nun interessierte, war, welche Art FSM in unserem Modellspiel Ökolopoly vorkommen können.

Wir haben festgestellt, dass wir eigentlich nur Moore-Automaten und Medwedjew-Automaten haben. Es gibt keine Eingänge, die direkt die Ausgänge verändern. Wir müssen aber zusätzlich zu den FSMs auch eine Klasse ausschliesslich für einen User-Input einplanen. Die Verteilung der Aktionspunkte zum Beispiel ist nämlich gar keine FSM. Je nach Eingang wird eine unbestimmte Verteilung auftreten. Der Ausgang ist somit nicht

abhängig vom Zustand und von den Eingängen, sondern ist ausschliesslich abhängig von der Entscheidung des Spielers. Man könnte natürlich argumentieren, dass dies eine degenerierte FSM sei, die in einem einzelnen Zustand verharrt ist. Da der Benutzer aber bei einer bestimmten Anzahl Aktionspunkte einmal eine Verteilung so, einmal anders definieren kann, widerspricht das aber der mathematischen Definition einer Funktion. Es gilt (*f: A -> Z*), wobei einem Element aus {A} nur immer genau ein einziges Bild aus der Menge {Z} zugeordnet werden darf. Und weil es sich um keine Funktion handelt, ist es also nicht als FSM realisierbar, die immer eine Funktion implementiert: zu jedem Zustand und Eingang gibt es nur einen einzigen klar definierten Ausgang.

Ebenso mussten wir die Frage lösen, wie wir die Ablaufsteuerung realisieren. Beim Spiel Ökolopoly ist die Reihenfolge, wann welche Scheibe gedreht wird, nämlich essentiell. Es ist deshalb wichtig, dass eine FSM erst dann ihre Eingänge verarbeitet, wenn die andere ihren Ausgang bereits eingestellt hat. Sonst verarbeitet eine FSM eine alte Information, was zu Inkonsistenzen führt. Die Idee, dass wir diese Steuerung z.B. mit einer Art Tokenring lösen können, war schnell geboren. Es kann so nur eine FSM gleichzeitig arbeiten, alle anderen müssen warten. Dadurch ist auch die Reihenfolge gesichert. Der Tokenring war aber zu wenig generisch, denn es ist nicht gesagt, dass wirklich alle FSM warten müssen. Zwei FSMs die sich gegenseitig nicht beeinflussen können, dürfen auch parallel arbeiten.

Für unser generischeres Konzept haben wir das Petri-Netz analysiert und als brauchbare Lösung empfunden. Wir haben gesehen, dass wir an den „Places" des Petrinetzes die FSMs anhängen und sie so steuern können. FSMs, die sich nicht beinflussen, können problemlos parallel arbeiten.

Bei der Implementierung des Petrinetzes sind wir aber zwei Problemen begegnet:

Erstens hatten wir bei einem zyklischen Petrinetz das Problem, dass jede Methode eine andere Methode aufruft und so immer ein neuer Stack pro Methode aufgebaut wird. Der Stack wird aber nie mehr abgebaut, da ständig jede Methode eine neue Methode aufruft, und so nie ein Rückgabewert (void) zurückfliessen kann. Eine Stackoverflow-Exception war so vorprogrammiert.

Lösung: Wir haben den „Places" und „Transition" des Petrinetzes ein Herz verpasst (Interface „Heart"). Die Objekte durften dann ihre Tokens erst verarbeiten, wenn sie von einer aussenstehenden Instanz einen Herzschlag bekommen haben (über die Methode „heartbeat()"). Während dem Herzschlag schauen sie nach, ob sie ein Token bekommen haben. Haben sie eines bekommen, dann werden sie das Token verarbeiten und evtl. an eine andere Instanz weiterleiten. Dazu müssen sie eine Methode dieser Instanz aufrufen. Das Erhalten eines Tokens führt aber nur zu einer Erhöhung der Tokens in der Instanz und nicht zu einer weiteren Aktion.

Die Instanz verarbeitet es erst, wenn es seinerseits einen Herzschlag erhalten hat. Die Heartbeats werden dann von einer einzelnen Methode eines zentralen Objektes ausgelöst, welche endlos Herzschläge sequentiell an die einzelnen Instanzen von „Transition" und „Places" verteilt. Der Stack wird nun nur noch für drei Methoden aufgebaut: Für die Steuerungsmethode, die Heartbeats verursacht, der Methode heartbeat() in der angestossenen Instanz und der getToken()-Methode der angeschlossenen Instanzen. Dann wird der Stack abgebaut und ein weiterer Heartbeat wird von der Steuerungsmethode ausgelöst. Wir haben das Problem also ohne Multithreading lösen können. Wir haben somit aber noch einen zusätzlichen Vorteil erreicht: Durch die Implementierung einer Basis-Methode ist nun auch die Steuerung der Abarbeitungsgeschwindigkeit möglich geworden. Beim Ökolopoly wird ja Wert darauf gelegt, dass der Spieler die Kette von Wirkungen und Rückwirkungen auch zu sehen bekommt. Das geht nur in einem vernünftigen Tempo.

Zweitens mussten wir eine Lösung finden, wie wir eine FSM zwei- oder dreimal ansprechen können, weil dies im Spiel Ökolopoly notwendig ist. Dort wird eine Ausgabe einer FSM wieder zur eigenen Eingabe. Ein Eingang A verursacht den ersten Zustandsübergang, welcher den zweiten Eingang B verändert. Der zweite Eingang B verursacht in einem weiteren Schritt einen neuen Zustandsübergang. Es ist wichtig, dass diese Reihenfolge eingehalten wird. Die FSM muss also „wissen" ob sie nun zum ersten oder zum zweiten Mal einen Zustandsübergang machen muss und somit den Eingang A oder den Eingang B verarbeiten wird.

Lösung: Die „Place" übergibt der FSM noch eine Referenz auf sich selber, wenn sie die FSM benachrichtigt. So lassen sich über mehrere „Places" dieselbe FSM ansprechen. Die FSM kann aber dank den Referenzen unterscheiden, ob sie nun von der ersten oder von der zweiten „Place" aufgerufen wird und nun das erste oder zweite Mal in einer Runde abgearbeitet wird.

7 Resultat

Unser Konzept für ein generisches Wirtschaftsspiel besteht grundsätzlich aus zwei Bestandteilen. Einerseits haben wir die FSMs, die die Funktionen des Planspiels abbildet und andererseits läuft übergeordnet ein Petrinetz ab, welches die Steuerung der einzelnen FSM übernimmt.

7.1 Finite State Machines

Ein endlicher Zustandsautomat kann man schematisch etwa so darstellen:

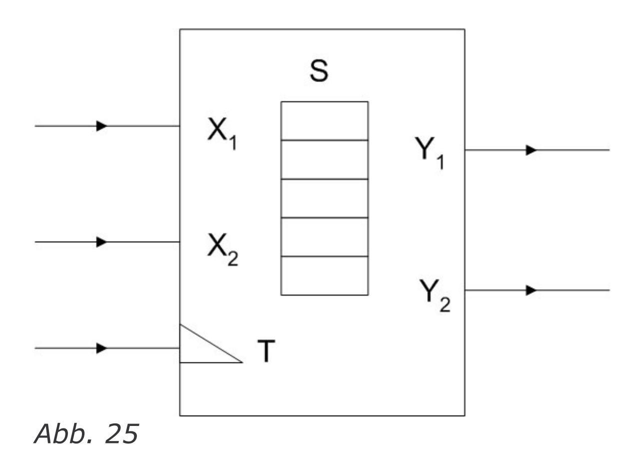


Abb. 25

Er hat Eingänge $x_1, x_2, ..y_n, t$ und Ausgänge $y_1, y_2 ... y_n$. Der Zustandsautomat befindet sich in einem Zustand, aus einer endliche Anzahl Zuständen. Bei den Eingängen ist „t" ein besonderer Eingang. (Bei uns handelt es sich hier um den Heartbeat.) Der Zustandsautomat muss nämlich ermitteln können, wann genau er seinen inneren Zustand ändern muss.

Mealy-Automat:

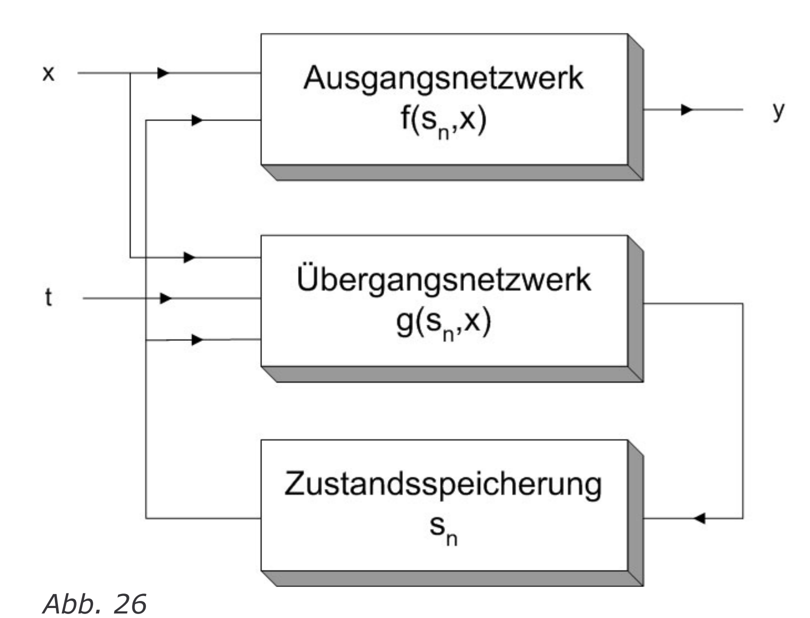


Abb. 26

Der Mealy-Automat ist der generellste aller FSMs.

Die Ausgänge sind eine Funktion der Eingänge und des inneren Zustandes. $Y_n = f_n(x,s)$ und der Zustand ist ebenso eine Funktion der Ausgänge und des inneren Zustandes. $S = g(x,s)$. Die Funktion ist in einer Tabelle abbildbar.

Beispiel:

Ein Ausgang lässt sich so ausdrücken: $Y_1 = a_{s1} * x_1 + b_{s1} * x_2 + c_{s1} * x_3$

Es sollen nun aber folgende Werte an diesem Ausgang auftreten:

Befindet sich der Zustandsautomat im Zustand 1, so ist $Y_1 = x_1$
Befindet sich der Zustandsautomat im Zustand 2, so ist $Y_1 = 2*x_1 - 3*x_2$
Befindet sich der Zustandsautomat im Zustand 3, so ist $Y_1 = x_2 + x_3$

Diese Ausgänge für Y_1 lassen sich als Funktion mit dieser Abbildungstabelle darstellen:

s	a_1	b_1	c_1
1	1	0	0
2	2	-3	0
3	0	1	1

So lässt sich anhand der einzelnen Parameter der Wert am Ausgang im entsprechenden Zustand berechnen.

Für jeden Ausgang wird dann also eine separate Abbildungstabelle benötigt.

Der Mealy-Automat ist aber sehr hypothetisch, da er in den einzelnen Planspielen eigentlich nicht vorkommt.

Beispiel Mealy-Automat als Lichttrimmer:

Es gibt bei einem Trimmer zwei Zustände: „Licht aus", „Licht ein". Wenn sich der Automat im Zustand „Licht ein" befindet, so kann über ein Potentiometer stufen- und zustandslos die Helligkeit verstellet werden. Somit wirkt ein Eingang direkt an den Ausgang.

Beispiel Mealy-Automat als Radio:

Ein weiteres Beispiel für einen Mealyautomat ist ein älteres Radio, wo die Lautstärke und die Frequenz noch über Potentiometer verstellt werden können. Die Grundwelle (UKW, Mittelwelle, Langwelle etc.) ist der Zustand des Radios. Die Frequenz und die Lautstärke sind aber stufenlos einstellbar. Es sind also Eingänge, die wieder direkt den Ausgang beinflussen. Eine Änderung des Eingangs bewirkt sofort eine Änderung des

Ausgangs. Neuere Radios haben aber einen Moore-Automat implementiert: Sowohl die Frequenz, wie auch die Lautstärke werden über eine Taste eingestellt. Ein Tastendruck ändert dann die Frequenz und die Lautstärke um eine bestimmte Grösse. Die Lautstärke und die Frequenz sind dann aber nur noch abhängig vom inneren Zustand des Radios.

Bei unserem näher untersuchten Planspiel Ökolopoly kommen keine Mealy-Automate vor. Ebenso ist das „New Commons Game" kein Mealy-Automat.

Moore-Automat:

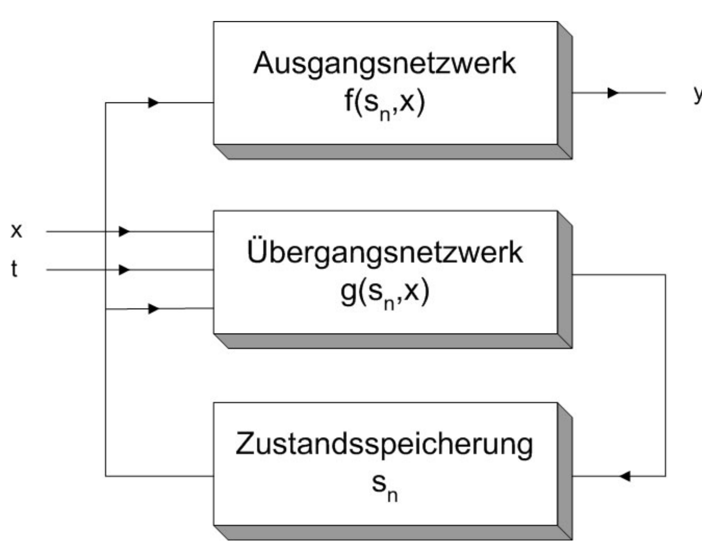


Abb. 27

Die Moore-Automaten sind einfacher konzipiert, weil dort die Ausgänge nur noch vom Zustand abhängig sind. Trotzdem bleibt es eine Funktion, da der Ausgang vom State berechenbar ist. y=f(s)

Beispiel:

$Y_1 = a_s$

Die Abbildungstabelle pro Ausgang kann dann so aussehen:

s	a
1	15
2	25
3	-5

Beim Spiel Ökolopoly kommt diese Form der Automaten am meisten vor. Z.B. die „Umweltverschmutzung" ist ein Moore-Automat mit drei Eingängen, zwei Ausgängen und 29 Zuständen.

Medwedjew-Automat:

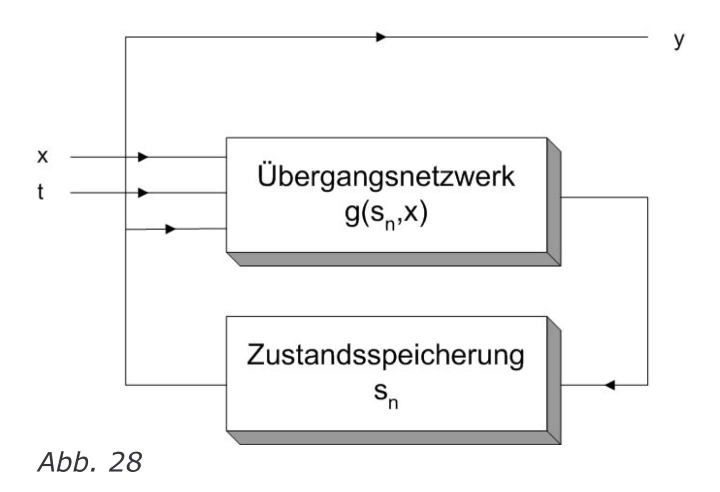


Abb. 28

Der Medwedjew-Automat ist noch einfacher als der Moore-Automat. Beim Medwedjew-Automaten entspricht der Wert am Ausgang dem Zustand. Es entfällt somit die Abbildungstabelle. Es zählt nur noch die Übergangstabelle.

Beispiel:

$Y_1 = s;$

Beim Spiel Ökolopoly sind die Aktionspunkte ein Beispiel für einen Medwedjew-Automat. Wenn die Scheibe Aktionspunkte auf dem Zustand 20 ist, dann sind auch 20 Aktionspunkte am Ausgang verfügbar.

Zustandsänderungsfunktion:

Die Zustandsänderung kann mit einer besonderen Abbildungstabelle und zwar mit einer Übergangstabelle dargestellt werden. Das Ganze ist ebenso eine Funktion, denn der neue Zustand errechnet sich dann aus den Eingängen und dem alten Zustand.

Beispiel:

$s_{n+1} = a_s*x_1 + b_s*x_2 + c_s*x_3 + d_s*s_n$

Wir haben beim Ökolopoly beobachtet, dass die Aenderung des Zustandes in Phasen ablaufen kann. D.h. zuerst bewirkt nur Eingang A eine Zustandsänderung. Ist der neue Zustand erreicht, dann bewirkt nur Eingang B eine Zustandsänderung. Das hat den Grund, weil der Ausgang Z gleich dem Eingang B sein kann. D.h. die Finite State Machine muss zuerst in einen neuen Zustand kommen, um überhaupt ermitteln zu können, wie sein Eingang B aussieht. Wir sind es aber gewohnt bei Finite State Machines meistens nur mit einem Uebergang pro Schritt zu rechnen. Die Lösung sieht aber so aus: unsere FSMs können nun mehrere „T"-

Eingänge haben. Bei „T 1“ wird nur Eingang A berücksichtigt, bei „T 2“ wird nur B berücksichtigt. Die Steuerung muss also zuerst „T 1“ erzeugen und beim nächsten Durchgang „T 2“. Wie wir das konkret lösen können, ist bereits in der Vorgehensweise ausführlich erläutert worden. (Seite 20)

7.2 Realisation Framework

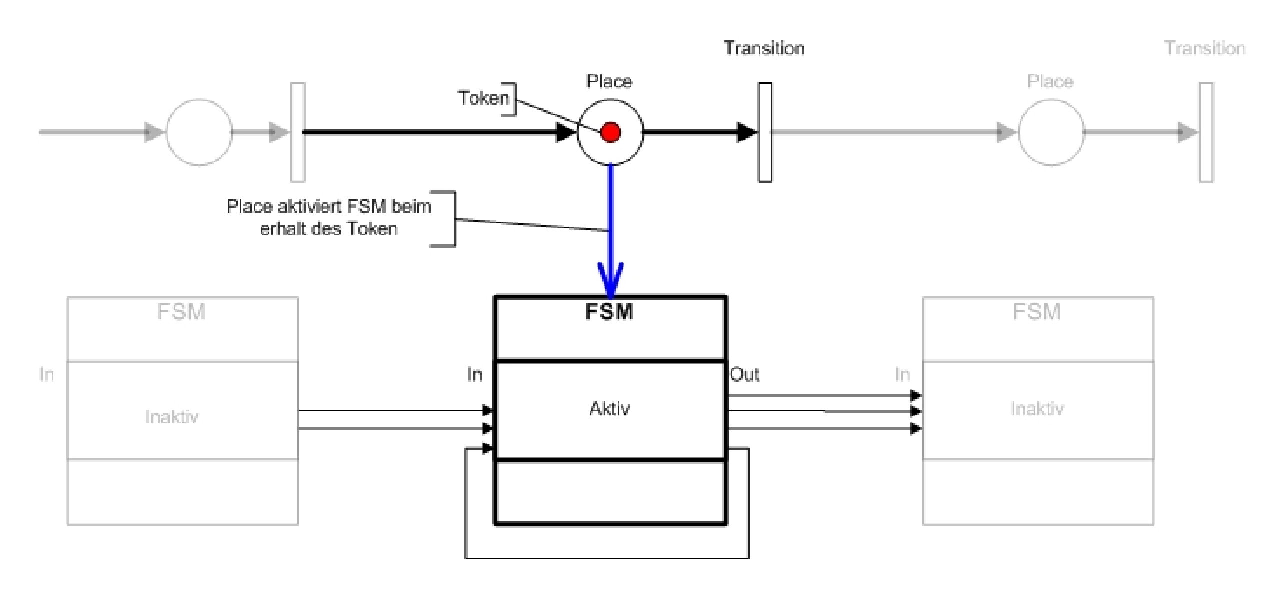


Abb. 29: Konzept des Frameworks

Die Verlaufssteuerung können wir mit einem Petri-Netz realisieren. Die „Place“ ist verantwortlich für Aktivierung einer FSM. Das geschieht immer dann, wenn sie von einer „Transition“ ein Token bekommt. Eine „Place“ kann das Token immer nur dann an einer „Transition“ weitergeben, wenn die FSM abgearbeitet worden ist. Die „Transition“-Instanzen sind dann für die eigentliche Verlaufssteuerung verantwortlich. Hat eine „Transition“ die Tokens aller an ihr angehängten „Places“ erhalten, so schickt sie das Token an alle registrierten „Places“ weiter. Eine Transition kann dazu das Token vermehren bzw. zu einem einzigen zusammenfassen.

Das Design der Implementierung des Petrinetzes sieht wie folgt aus:

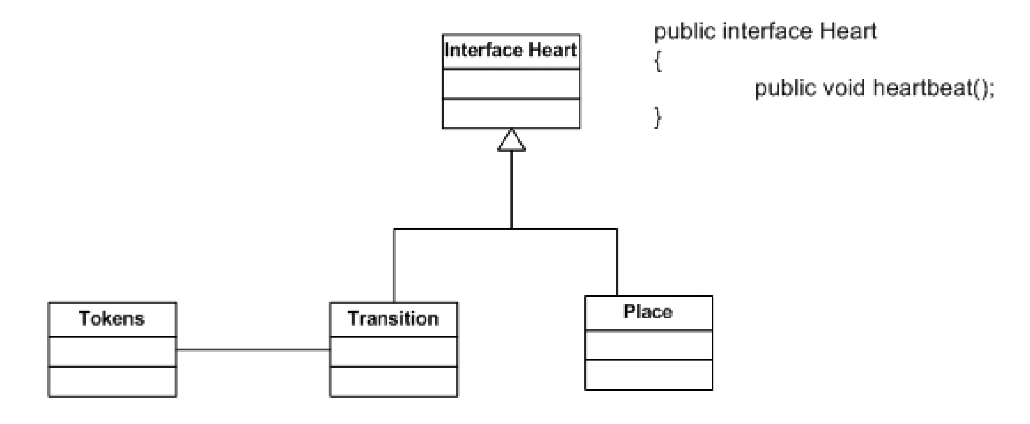


Abb. 30: Design Petrinetz

Die Klassen Transition und Place implementieren das Interface Heart, welches aus der „heartbeat()“-Methode besteht.

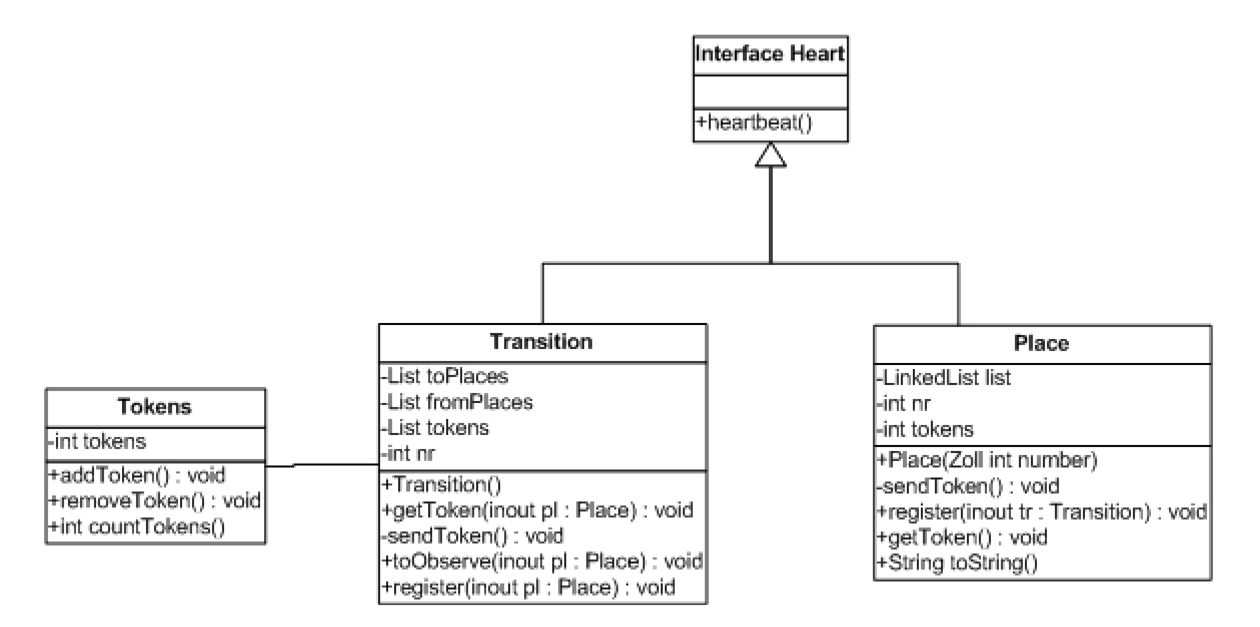


Abb. 31: Design Petrinetz

7.2.1 Implementierung in Java

```
public class Place implements Heart
{
	List list = new LinkedList();	// merkt sich die Transitions
	int nr;				// ist die ID der Instanz
	int tokens=0;			// merkt sich die Anzahl Tokens.

	public Place(int number)
	{
		nr=number;
	}

	// Methode zum Registrieren von einer Transitions-Instanzen
	public void register(Transition trans)
	{
		list.add(trans);
	}

	//Methode, dass die Place-Instanz ein Token zufällig an eine regisitrierte
	// Transition-Instanz weiterleitet.
	private void sendToken()
	{
		if(tokens>0)
		{
			Iterator itr = list.iterator();

			if (itr.hasNext())
			{
				Object[] array = list.toArray();

	((Transition)array[(int)(Math.random()*array.length)]).getToken(this);
			}
			tokens--;
		}


	}

	//Methode um ein Token zu erhalten und die Anzahl Tokens zu erhöhen.
	public void getToken()
	{
		System.out.println("Place "+nr);
		tokens++;
	}

	//Das implementierte Interface "Heart"
	//Die Methode lost das Senden eines Tokens aus.
	public void heartbeat()
	{
		sendToken();
	}

}
```

```
public class Transition implements Heart
{

        List toPlaces = new LinkedList();         //merkt sich, an welche Places die Tokens
                                                  //weitergeleitet werden müssen.
        List fromPlaces = new LinkedList();       //merkt sich, von welchen Places es Tokens
                                                  //erhält in einer separaten Queue.
        List tokens = new LinkedList();           // merkt sich die Tokens pro Place.
        int nr;                                   // die Id der Instanz¨.

        public Transition(int number).
        {
                nr=number;
        }

        // Das implementierte "Heart"-Interface
        public void heartbeat()
        {
                sendToken();
        }

        // Methode um Place-Instanzen zu registrieren.
        public void register(Place pl)
        {
                toPlaces.add(pl);
        }

        // Methode, um Place-Instanzen zu registrieren um zu identifizieren von welcher
        // Place-Instanz die Transition Tokens erhalten hat.
        public void toObserve(Place pl)
        {
                fromPlaces.add(pl);
                tokens.add(new Tokens());
        }

        // die Methode überprüft, ob sie von allen registrierten Zulieferer-Place-Instanzen
        // je  ein Token  bekommen hat. Wenn dies erfüllt ist, dann kopiert sie das Token
        // so  häufig, wie es registrierte Abnehmer-Place-Instanzen hat und sendet jeder
        // Instanz ein Token zu. Und sie löscht bei sich dann bei jeder Queue ein Token.
        private void sendToken()
        {
                boolean test = true;
                Iterator itr = tokens.iterator();
                while(itr.hasNext())
                {
                        if (((Tokens)itr.next()).countTokens()==0) test=false;
                }
                if (test)
                {
                        itr = tokens.iterator();
                        while(itr.hasNext())
                        {
                                ((Tokens)itr.next()).removeToken();
                        }

                        itr = toPlaces.iterator();
                        while(itr.hasNext())
```

```
                {
                        ((Place)itr.next()).getToken();
                }
        }
}

// ist eine Methode um ein Token zu erhalten. Dabei identifiziert es das Zulieferer-
// Place-Objekt und ordnet es in die richtige Queue ein.
public void getToken(Place pl)
{
        System.out.println("Transistion "+nr+"  got Token from: "+ pl.toString());
        Iterator plitr = fromPlaces.iterator();
        Iterator tokitr = tokens.iterator();

        while (plitr.hasNext())
        {
                Tokens tokens = (Tokens)tokitr.next();
                if (((Place)plitr.next())==pl)
                {
                        tokens.addToken();
                }
        }
}

// ist eine Klasse, die von der Transition benötigt wird. Sie stellt eine Art Queue
// dar. wo Tokens eingereiht werden. Diese Objekte werden in der Transition in
// einer Liste  geführt. Es exisitiert eine Instanz dieser Klasse pro registrierte
// Zulieferer-Place-Instanz.
class Tokens
{
        int tokens=0;

        // Methode um ein Token aus der Queue zu entfernen.
        public void removeToken()
        {
                tokens--;
        }

        // Methode um ein ein Token zur Queue hinzuzufügen.
        public void addToken()
        {
                tokens++;
        }

        // Methode, die Auskunft gibt, wieviele Tokens in der Queue drin sind.
        public int countTokens()
        {
                return tokens;
        }
}

}
```

8 Unser Konzept am Beispiel von Ökolopoly

Wir wenden nun unser Konzept am Spiel Ökolopoly an.

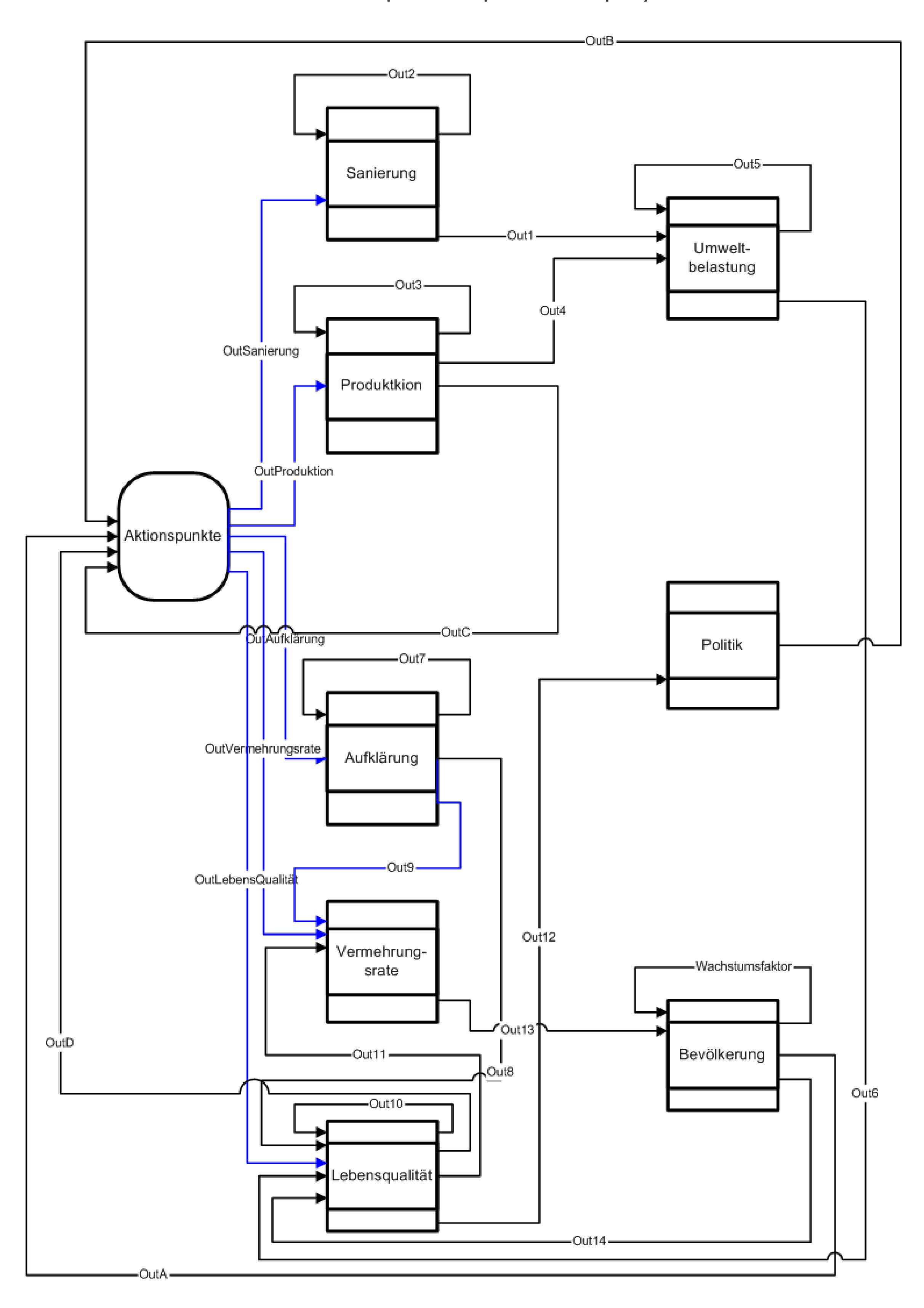


Abb. 32: Ökolopoly abgebildet auf die verschiedenen Automaten

Als erstes haben wir das Planspiel mit der Idee der Aufteilung auf mehrere Automaten aufgezeichnet.
In Abb. 33 werden die einzelnen Bereiche im Spiel als Automaten mit ihren Abhängigkeiten dargestellt. Die blauen Verbindungen zeigen, wo eine Benutzereingabe nötig ist. Beispielsweise werden am Anfang jeder Runde die Aktionspunkte verteilt. Die Verteilung der Aktionspunkte ist nicht als Automat realisiert. Die Ausgänge wurden fortlaufend nummeriert mit "Out"+Nr Analog zum Brettspiel.

Später haben wir uns die Verlaufssteuerung mit dem Petrinetz für dieses Planspiel überlegt.

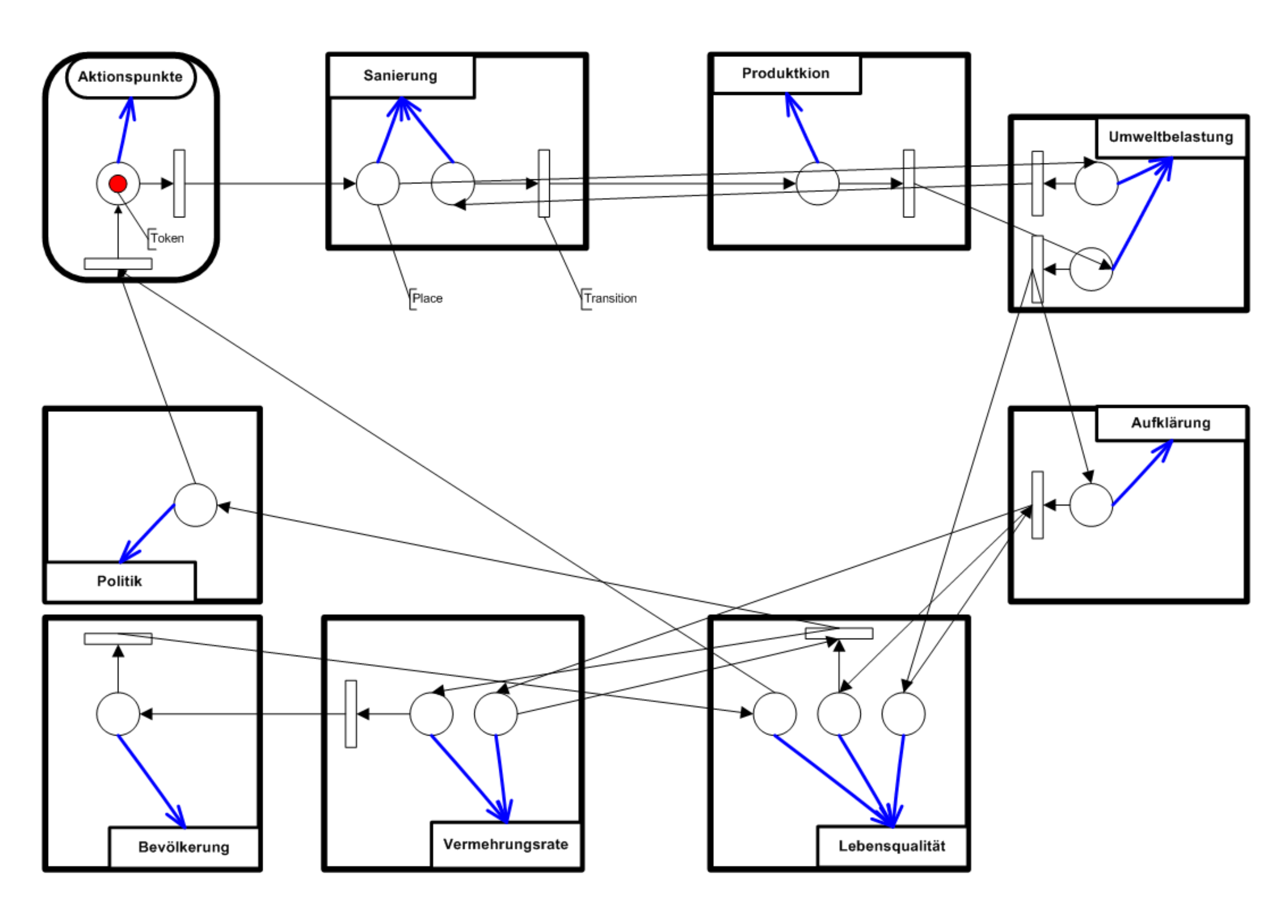


Abb. 33: Verlaufsteuerung mit dem Petrinetz

Die Automaten sind als Kasten, wie auf dem Brettspiel angeordnet. Zusätzlich haben wir jetzt das Petrinetz darüber gespannt. Gelangt das rote Token in einem „Place" aktiviert dieser den Automaten (blauer Pfeil). Anzumerken ist hier, dass einem Bereich, der während einer Spielrunde mehrmals verändert wird, mehrere „Places" zugewiesen werden.

Hier noch ein Ausschnitt aus dem Spiel mit allen Verknüpfungen:

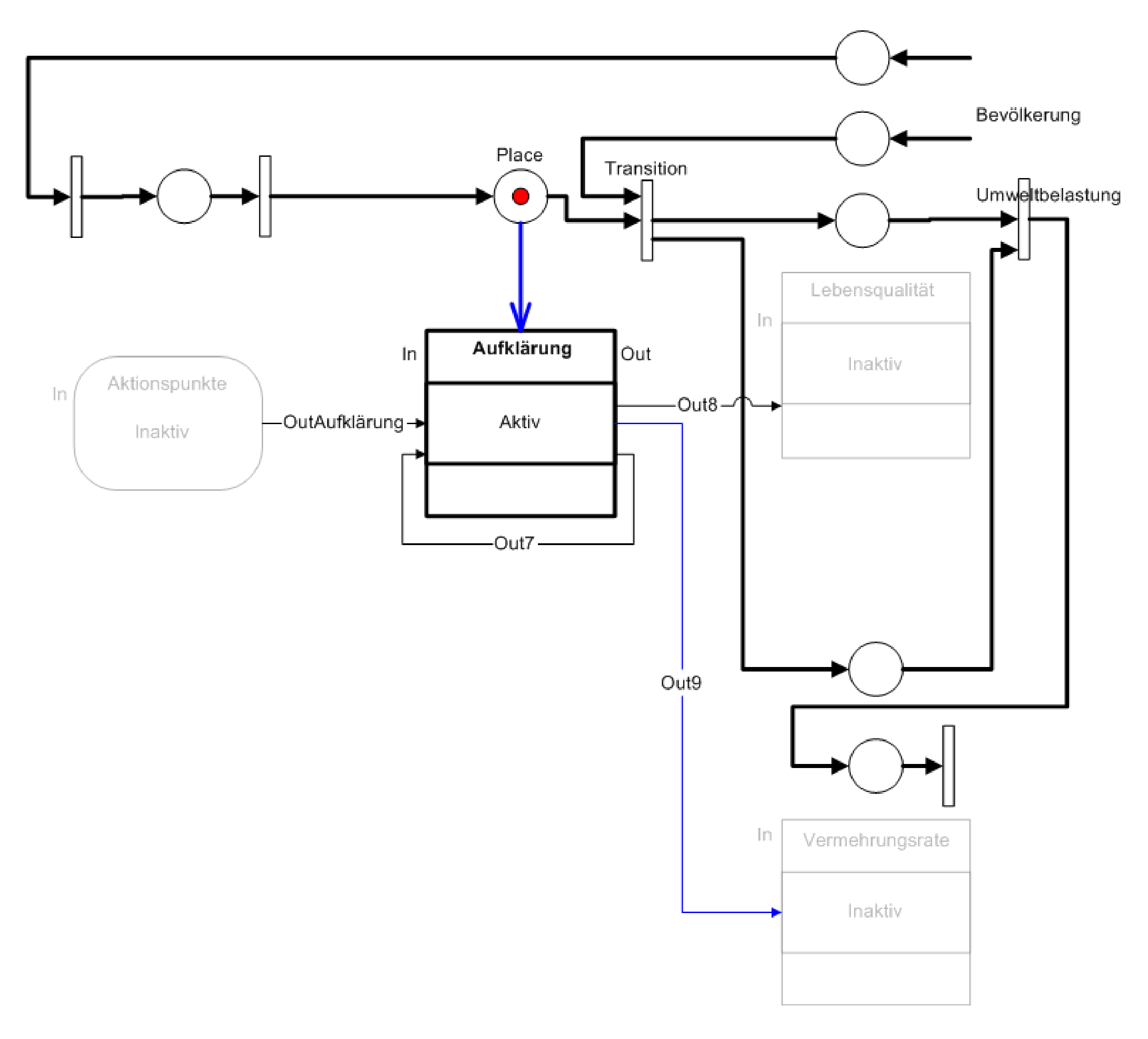


Abbildung 34: Ausschnitt aus dem Ökolopoly

9 Quellenverzeichnis

[1] Schmid, Elisabeth und Stutzter Tutti: Generisches Planspiel, Diplomarbeit ZHW, 2004.

[2] Ulrich, Markus: Mit Planspielen nachhaltig Wirkung erleben, in: www.ucs.ch/service/download/docs/artikelpsnaha.pdf 18.2.2005

[3] Ulrich, Markus: Sind Planspiele langwierige und kompliziert, in: www.vernetzt-denken.de/PDF_Dokumente/5_01.pdf 18.2.2005

[4] Capaul, Roman und Ulrich, Markus: Planspiele, Altstätten Schweiz(Tobler Verlag AG), 2003

[5] Ulrich, Markus: Über Planspiele, in: www.ucs.ch/planspiele/ueber/index.html 18.2.2005

[6] Ravensburger: Ökolopoly Informationsheft Otto Maier Verlag Ravensburger GmbH, 1993

[7] Vester, Frederic: Die Kunst vernetzt zu denken, München (Deutscher Taschenbuch Verlag GmbH & Co. KG), 2003

[8] Vester, Frederic: Neuland des Denkens München (Deutscher Taschenbuch Verlag GmbH & Co. KG), 2002

[9] Siemers, Christian und Sikora, Axel: Taschenbuch Digitaltechnik, München (Carl Hanser Verlag), 2003

[10] TU Darmstadt: Glossar der Spielpädagogik, in: www.tu-darmstadt.de/hjd/spielpae/glossar.htm 18.2.2005